УДК 004.62

# БЫСТРОДЕЙСТВУЮЩИЙ АЛГОРИТМ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ

М.П. Шарабайко, А.Н. Осокин

Томский политехнический университет
E-mail: sme_box@tpu.ru

*Исследованы наиболее эффективные с точки зрения уменьшения времени сжатия модификации алгоритма фрактального сжатия изображений. На основе исследования программных реализаций синтезирован быстродействующий алгоритм фрактального сжатия, использующий классификацию блоков, распараллеливание процесса сжатия, переупорядочивание пикселей блоков в памяти. Данные модификации существенно ускоряют процесс сжатия (до 1000 раз по сравнению с базовым) при сохранении размера сжатого файла и несущественных потерях в качестве. Модификация разбиения методом квадродерева упрощает процесс разбиения и работу с изображениями неквадратных размеров.*

***Ключевые слова:***
*Фрактальное сжатие изображений, классификация блоков, классификация Фишера, классификация на основе центра масс, классификация Саупе, классификация Саупе–Фишера, разбиение квадродеревом.*

***Key words:***
*Fractal image compression, block classification, Fisher classification, Saupe classification, mass-center classification, quadtree partitioning.*

Фрактальное сжатие изображений – алгоритм сжатия с потерями, основанный на представлении изображения в более компактной форме с помощью коэффициентов систем итерируемых кусочно-определённых функций (PIFS – Partitioned Iterated Function Systems), как правило, являющихся аффинными преобразованиями частей изображения [1]. Степень сжатия изображений может достигать 100:1 [2].

Фрактальная компрессия стала практически реализуемой после введения Арно Жаквином (Arnaud Jacquin) [3] понятия итерируемых кусочно-определённых функций, в которых, каждое из набора отображений покрывает изображение частично, а не целиком.

На текущий момент основными недостатками алгоритма являются большие временные затраты сжатия и невозможность гарантировать ту или иную степень потерь (качество декодированного изображения зависит от самоподобия сжимаемого). Достоинства включают степень сжатия на уровне JPEG при сравнительно одинаковом качестве, быстрый процесс декодирования, независимость восстанавливаемого изображения от разрешения (хранится структура изображения, а не данные о пикселях), потери проявляются в виде размытия изображения, а не в виде высокочастотных шумов в области контрастных переходов, свойственный алгоритму JPEG.

Характеристики современных ЭВМ позволяют преодолеть проблему скорости сжатия, сохраняя перечисленные достоинства. Последние могут найти широкое применение в области сжатия видеоинформации, например, в технологии Intel Wireless Display [4].

Целью работы является исследование имеющихся модификаций алгоритма и их объединение для увеличения быстродействия базового.

В общих чертах, фрактальное сжатие можно разделить на два этапа:

1) разбиение изображения на множество ранговых блоков и на множество доменных блоков (которые могут перекрывать друг друга);
2) применение преобразований для каждой пары доменный–ранговый блок: геометрическое, отображающее доменный блок в ранговый, и аффинное, изменяющее значения яркости доменного блока до максимального соответствия значениям яркости рангового блока.

От схемы разбиения, используемой на первом этапе, зависит качество сжатия. Чем больше доменных блоков, тем больше шанс найти наиболее подобный ранговому блок.

На втором этапе необходимо так преобразовать доменный блок, чтобы он был максимально подобен ранговому. Общая формула преобразования значений пикселей доменного блока выглядит следующим образом:

$$D_i^* = sD_i + o, \qquad (1)$$

где $D_i^*$ и $D_i$ – преобразованный и исходный $i$-й доменный блок соответственно; $s$ – коэффициент изменения контраста; $o$ – коэффициент сдвига по яркости.

Помимо непосредственного преобразования значений пикселей по формуле (1), доменный блок также может быть подвергнут общему масштабированию (уменьшение размеров до размеров рангового блока, например, интерполяцией или простым прореживанием), повороту и другим аффинным преобразованиям.

Преобразованный доменный блок должен соответствовать ранговому блоку как можно сильнее, поскольку именно так ранговый блок будет восстановлен при декодировании. Для оценивания расхождения (расстояния) между преобразованным доменным и данным ранговым блоками, необходимо ввести соответствующую метрику. Обычно используется функция среднеквадратического отклонения (СКО) [1, 2, 5]:

$$Q = \sum_{i=1}^{N} (D_i^* - R_i)^2 = \sum_{i=1}^{N} ((sD_i + o) - R_i)^2, \qquad (2)$$

где $R_i$ – $i$-й ранговый блок, $D_i^*$ и $D_i$ – преобразованный и исходный $i$-й доменный блок соответственно, $N$ – количество пикселей в ранговом блоке.

Очевидно, чем меньше расстояние (2) между блоками, тем больше они подобны.

Коэффициенты $s$ и $o$ можно найти из формулы (2), взяв частные производные по этим переменным.

Раскроем квадрат в выражении (2):

$$Q = \sum_{i=1}^{N} (s^2 D_i^2 + 2soD_i + o^2 - 2sR_iD_i - 2oR_i + R_i^2). \quad (3)$$

Имеем следующее условие:

$$\begin{cases} \dfrac{\partial Q}{\partial s} = \sum\limits_{i=1}^{N} (2sD_i^2 + 2D_io - 2R_iD_i) = \\ \quad = s\sum\limits_{i=1}^{N} D_i^2 + o\sum\limits_{i=1}^{N} D - \sum\limits_{i=1}^{N} R_iD_i = 0; \\ \dfrac{\partial Q}{\partial o} = \sum\limits_{i=1}^{N} (sD_i + o - R_i) = 0. \end{cases} \quad (4)$$

Выразим сдвиг по яркости:

$$o = \frac{1}{N}\sum_{i=1}^{N} R_i - \frac{1}{N} s\sum_{i=1}^{N} D_i. \quad (5)$$

Подставим (5) в уравнение частной производной по $s$ (4) и получим следующие формулы для нахождения коэффициентов:

$$\begin{cases} s = \dfrac{N\sum\limits_{i=1}^{N} R_iD_i - \sum\limits_{i=1}^{N} R_i \sum\limits_{i=1}^{N} D_i}{N\sum\limits_{i=1}^{N} D_i^2 - (\sum\limits_{i=1}^{N} D_i)^2}; \\ o = \dfrac{1}{N}(\sum\limits_{i=1}^{N} R_i - s\sum\limits_{i=1}^{N} D_i). \end{cases} \quad (6)$$

Преобразовав формулу (3), получим выражение для нахождения расстояния:

$$Q = s^2\sum_{i=1}^{N} D_i^2 + No^2 + \sum_{i=1}^{N} R_i^2 - \\ -2s\sum_{i=1}^{N} R_iD_i + 2so\sum_{i=1}^{N} D_i - 2o\sum_{i=1}^{N} R_i. \quad (7)$$

Формулы (6, 7) позволяют упростить вычислительную нагрузку, так как суммы $\sum\limits_{i=1}^{N} R_i$, $\sum\limits_{i=1}^{N} R_i^2$, $\sum\limits_{i=1}^{N} D_i$, $\sum\limits_{i=1}^{N} D_i^2$ можно вычислить еще до начала перебора, когда уже сформированы множества ранговых и доменных блоков. Тогда на этапе сопоставления потребуется вычислить лишь сумму $\sum\limits_{i=1}^{N} R_iD_i$ и найти коэффициенты.

#### Порядок проведения исследований

Исследуемые модификации алгоритма реализуются программно на языке C++ для получения практически обоснованных характеристик (время сжатия, качество сжатия по метрикам SSIM и PSNR, размер сжатого файла, скорость декодирования).

Для правильного сравнения результатов испытания проводились с помощью одних и тех же аппаратных и программных средств.

Для оценки качества восстановленного изображения используется программа IMQ Ver. 110113[6].

Характеристики ПЭВМ: процессор Intel Core i3 530 2,93 ГГц, ОЗУ 2 Гб DDR3, ОС Windows 7. Индекс производительности Windows 5,5. Для тестов используются изображения в оттенках серого (8 бит/пиксель) группы фрактального кодирования и анализа (*fractal coding and analysis group*) [7].

Используемые далее характеристики включают в себя полное время работы алгоритма сжатия, алгоритма декодирования, размер сжатого файла, качество по метрикам SSIM и PSNR.

#### Выбор метода разбиения

Метод разбиения изображения на множество доменных и ранговых блоков накладывает ограничения на их размер и форму, и непосредственно влияет на качество и коэффициент сжатия. Известно достаточно большое количество различных схем построения системы ранговых блоков, среди них разбиение на множество блоков фиксированного размера, разбиение методом квадродерева, разбиение при помощи триангуляции.

В таблице 1 приведены результаты исследования перечисленных схем разбиения.

***Таблица 1***. *Результаты испытаний методов разбиения на изображении lena с разной глубиной разбиения*

| Алгоритм разбиения | Время сжатия, с | Время декодирования, с | Размер файла, кб | SSIM | PSNR |
|---|---|---|---|---|---|
| Базовый | 6,94 | 0,08 | 61,5 | 0,946 | 35,451 |
| | 0,81 | 0,08 | 14,4 | 0,849 | 29,795 |
| Квадродеревом | 7,05 | 0,10 | 50,8 | 0,936 | 33,356 |
| | 1,12 | 0,09 | 12,9 | 0,832 | 28,611 |
| Триангуляционный | 16,77 | 0,27 | 25,3 | 0,842 | 28,836 |
| | 89,45 | 0,35 | 40,7 | 0,893 | 31,746 |

Наиболее успешным оказалось разбиение методом квадродерева. К достоинствам метода можно отнести адаптивность: изображение разбивается на большее количество блоков на тех участках, где имеется больше деталей для кодирования.

Однако такое разбиение обладает и рядом недостатков. Фактически непосредственный процесс перебора блоков имеет смысл выполнять лишь на уровнях квадродерева с размерами ранговых блоков 16×16, 8×8, 4×4, 2×2 пикселя. То есть рекурсивно разбивать изображение, начиная с исходных размеров, нет необходимости: рекурсивное выполнение функции разбиения до достижения необходимого уровня является неэффективным и имеет ряд недостатков:

- замедляет процесс сжатия;
- усложняет распараллеливание алгоритма;

- ограничивает размер изображения: в качестве начального уровня выбирается наибольший квадрат, вписываемый в размеры изображения; оставшаяся часть, при ее наличии, либо игнорируется, либо обрабатывается отдельно.

Абсолютное большинство программных реализаций [8–13] использует данную схему разбиения, считая вершиной квадродерева наибольший квадрат, вписываемый в исходное изображение.

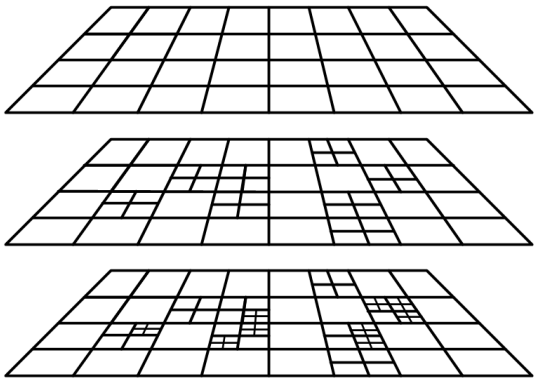

***Рис. 1.*** *Уровни разбиения изображения методом псевдоквадродерева*

От перечисленных выше недостатков можно избавиться, немного модифицировав подход, ориентируясь на эффективность программной реализации и сохраняя преимущества разбиения методом квадродерева. Далее в работе метод будет именоваться разбиением методом псевдоквадродерева, рис. 1.

Изображение можно сразу разбивать на множество блоков целевого размера, например, 16×16 пикселей (рис. 1, верхняя плоскость). И далее, если понадобится, дополнительно разбивать эти блоки (рис. 1, средняя и нижняя плоскости). Такой подход позволит освободиться от лишних операций рекурсивного разбиения изображения, хранения стека вызова функций, а также позволит кодировать изображение размером, кратным максимальному размеру доменного блока, аналогично другим алгоритмам сжатия изображения, например, JPEG, работающему с изображениями размерами, кратными 16 пикселям (до кодирования недостающие пиксели добавляются к области кодирования).

#### Переупорядочивание блоков изображения в памяти

Дополнительно ускорить работу алгоритма можно за счет подготовки памяти к процессу перебора блоков. Легче всего ускорить работу с памятью при кодировании для изображения, разбитого на блоки фиксированного размера.

Перед началом кодирования (доменно-ранговые сопоставления) пиксели изображения можно переупорядочить так, чтобы с точки зрения организации ОЗУ доступ к ним происходил максимально быстро.

На рис. 2, *а*, приведена схема обычного способа хранения изображения в оперативной памяти компьютера. Строки идут друг за другом, т. е. пиксели лежат последовательно в виде цепочки. Группы пикселей, принадлежащие одному блоку, выделены тоном. Поскольку при кодировании обращение происходит не к пикселям всего изображения, а к пикселям его блоков, то, переупорядочив данные в памяти, можно добиться последовательного обращения к ней же. Например, на рис. 2, *а*, изображение состоит из четырех блоков: 1–4; 5–8; 9–2 и 13–16. Пиксели одного блока выделены тоном. Поскольку информация в ОЗУ читается последовательно, то при чтении пикселей 1, 2, 3, 4 также будут считаны пиксели 5 и 6. Если переупорядочить изображение в памяти так, как показано на рис. 2, *б*, то процесс чтения оптимизируется и, как следствие, будет происходить быстрее.

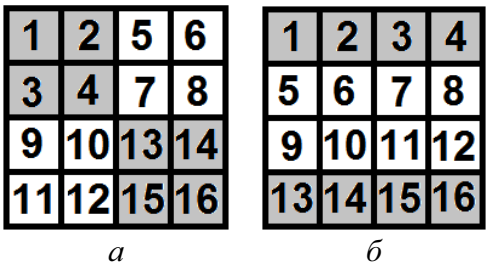


***Рис. 2***. *Вид хранения пикселей изображения в памяти компьютера: а) обычный; б) переупорядоченный*

Таким образом, создав два буфера переупорядоченного хранения ранговых и доменных блоков и потратив немного времени на копирование, можно существенно ускорить процесс сжатия, табл. 2.

***Таблица 2***. *Результат переупорядочивания пикселей блоков в памяти*

| Изображение | Время сжатия, с | | SSIM | PSNR |
|---|---|---|---|---|
| | Без переупорядочивания | С переупорядочиванием | | |
| Lena (512) | 2,92 | 1,48 | 0,834 | 28,850 |
| Frymire (1024) | 58,31 | 39,35 | 0,543 | 13,587 |

Подобное ускорение можно также применить для кодирования изображения, разбитого на блоки построением квадродерева. По заранее известным (вычисленным) данным о размере доменных блоков и шагу их выбора, можно подготовить доменный пул для каждого уровня квадродерева. В случае разбиения предлагаемым методом псевдоквадродерева, количество уровней существенно уменьшается за счет отбрасывания ненужных начальных.

Допустим, имеем три уровня разбиения: 16×16, 8×8 и 4×4 пикселей. Размеры доменных блоков будут соответственно 32×32, 16×16 и 8×8 пикселей. Если шаг выбора доменных областей совпадает с их размером, то понадобится в четыре раза больше памяти, чем при обычном подходе, когда хранится лишь непосредственно само изображение. Так, 256×256 пикселей изображения в оттенках се-

рого занимают 64 кбайт. Для хранения трех уровней доменного пула используется 256 кбайт, что для текущих характеристик ПЭВМ проблемой не является, а результат, как показано в табл. 2, заметный: ускорение почти в два раза без использования схем классификации и другого рода алгоритмической оптимизации.

#### Классификация блоков

Классификация доменных и ранговых блоков предназначена для уменьшения перебора блоков, и, как следствие, ускорения алгоритма сжатия. Каждый доменный блок классифицируется до начала кодирования. Во время подбора потенциальный ранговый блок также классифицируется и сравнивается только с доменами соответствующего класса (либо нескольких близких классов).

Наиболее эффективными признаются классификации [14]:

- Фишера (Y. Fisher) [2];
- на основе нахождения «центра масс» [15];
- Хёртджена (B. Hurtgen) [16];
- Саупе (D. Saupe) [17];
- Саупе–Фишера [2].

Результаты сравнительных испытаний, проведенных в идентичных условиях, приведены в табл. 3.

***Таблица 3.*** *Результаты испытаний методов классификаций блоков*

| Метод классификации | СКО | Время сжатия, с | Размер файла, кб | SSIM | PSNR |
|---|---|---|---|---|---|
| Изображение lena 512×512 | | | | | |
| Полный перебор | 8 | 1138,86 | 14,71 | 0,881 | 33,893 |
| | 2 | 3168,26 | 50,44 | 0,961 | 37,712 |
| Фишера | 8 | 0,61 | 16,51 | 0,874 | 33,257 |
| | 2 | 1,27 | 52,39 | 0,948 | 35,917 |
| Центр масс | 8 | 1,02 | 15,92 | 0,876 | 33,489 |
| | 2 | 2,23 | 51,94 | 0,951 | 36,440 |
| Хёртджена | 8 | 1,15 | 16,10 | 0,875 | 33,397 |
| | 2 | 2,55 | 52,25 | 0,949 | 36,258 |
| Саупе | 8 | 3,55 | 14,81 | 0,880 | 33,851 |
| | 2 | 12,49 | 50,84 | 0,961 | 37,642 |
| Саупе–Фишера | 8 | 0,91 | 15,22 | 0,879 | 33,742 |
| | 2 | 2,28 | 51,46 | 0,959 | 37,387 |
| Изображение frymire 1024×1024 | | | | | |
| Полный перебор | 8 | 25386,59 | 163 | 0,932 | 20,857 |
| | 2 | 31031,18 | 168 | 0,943 | 20,986 |
| Фишера | 8 | 53,19 | 161 | 0,921 | 20,293 |
| | 2 | 60,03 | 168 | 0,925 | 20,316 |
| Центр масс | 8 | 152,06 | 159 | 0,925 | 20,462 |
| | 2 | 184,01 | 167 | 0,929 | 20,479 |
| Хёртджена | 8 | 150,13 | 160 | 0,929 | 20,687 |
| | 2 | 163,30 | 167 | 0,933 | 20,717 |
| Саупе | 8 | 130,76 | 159 | 0,919 | 20,278 |
| | 2 | 149,39 | 167 | 0,922 | 20,296 |
| Саупе–Фишера | 8 | 21,69 | 161 | 0,882 | 18,819 |
| | 2 | 23,62 | 168 | 0,885 | 18,831 |

Классификация Саупе–Фишера показала очень хороший результат на обоих тестовых изображениях, используемых в практических испытаниях. Исследования на изображении большего разрешения (frymire, 1024×1024) позволяет говорить о существенном сокращении времени кодирования по сравнению со всеми остальными схемами классификации. Декодированное изображение при этом довольно близко к оригиналу, метод незначительно ухудшает визуальное сходство, оцениваемое по метрикам SSIM и PSNR. Визуально различия почти незаметны. Вторым по успешности методом классификации можно по праву считать схему Фишера.

#### Распараллеливание

В [18] показана возможность распараллеливания процесса фрактального сжатия изображений, а также линейная зависимость такого ускорения от числа процессорных ядер. Так, на двух ядрах имеет место двукратное ускорение. На одноядерных процессорах Intel с поддержкой технологии HyperThreading достигается 1,4 кратное ускорение.

#### Предлагаемая схема фрактального сжатия изображений

На основе проведенных исследований и полученных результатов, наиболее эффективной с точки зрения соотношения времени сжатия, времени декодирования, качества восстановленного изображения по метрикам SSIM и PSNR и размеру файла сжатого изображения будет схема фрактального сжатия изображений с разбиением методом псевдоквадродерева и классификацией блоков по схеме Саупе–Фишера с распараллеливанием и переупорядочиванием блоков. Блок-схемы сжатия и декодирования приведены на рис. 3, результаты испытаний – в табл. 4.

Итоговое тестирование быстродействующей схемы показало, что на фотореалистических изображениях, где характер потерь при сжатии алгоритмом JPEG проявляется менее заметно, фрактальное сжатие слегка отстает. На растровых изображениях геометрических фигур (изображения crosses, circles) фрактальное сжатие заметно выигрывает у JPEG, что обусловлено более высокой степенью самоподобия указанных изображений. Алгоритм JPEG хорошо справляется с прямоугольниками и линиями, расположенными вертикально либо горизонтально (изображения squares, horizon). На сжатии текстовых изображений алгоритм работает плохо (изображение text).

#### Выводы

Предложен быстродействующий алгоритм фрактального сжатия изображений, объединяющий в себе наиболее эффективные модификации базовой схемы. Классификация блоков, существенно снижающая количество переборов, переупорядочивание пикселей блоков а также распараллеливание процесса сжатия приводят к его ускорению. На начальном уровне модификация разбиения методом псевдоквадродерева сразу работает с блоками целевого размера (16×16 вместо размера сжимаемого изображения, например,

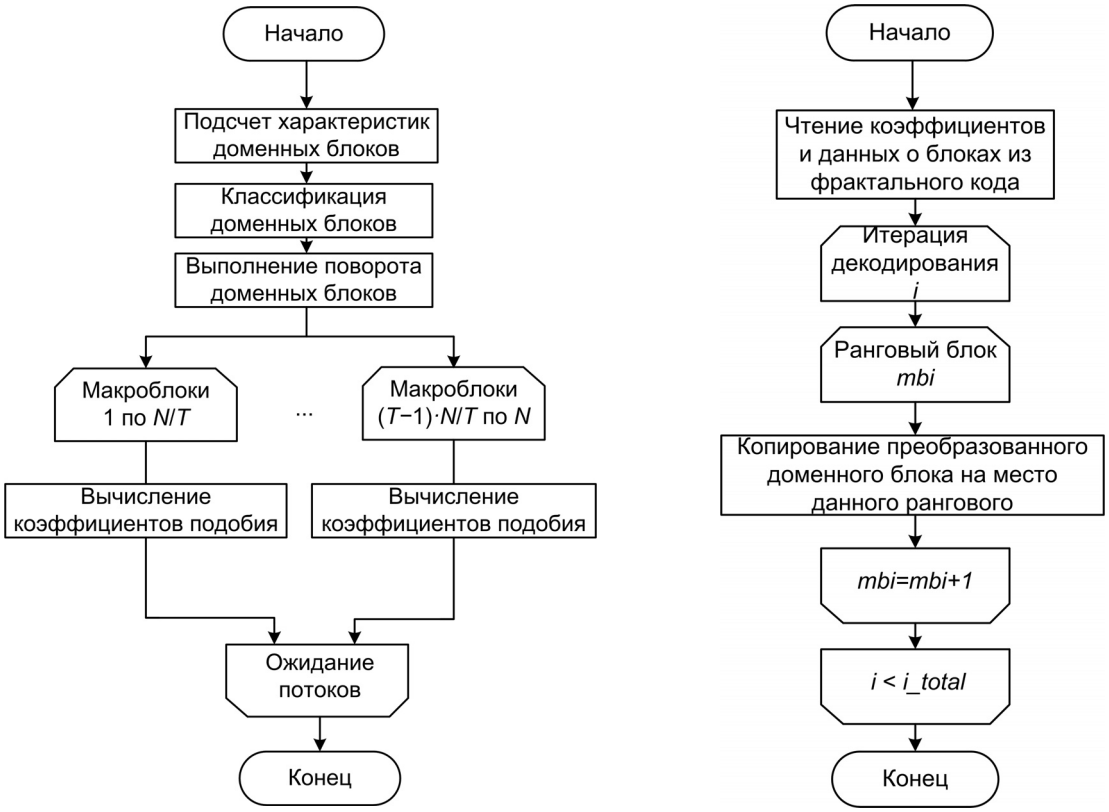


***Рис. 3***. *Блок схема комплексного алгоритма фрактального сжатия и декодирования изображений: а) сжатия; б) декодирования*

***Таблица 5.*** *Результаты испытания предлагаемого алгоритма*

| Изображение | Время сжатия, с | Полный перебор, с | Время декодера, с | Размер файла, кб | SSIM | PSNR | Размер файла jpg, кб | JPEG SSIM | JPEG PSNR |
|---|---|---|---|---|---|---|---|---|---|
| Lena | 0,511 | 1138,86 | 0,040 | 15,200 | 0,8773 | 33,7358 | 15,480 | 0,897 | 34,398 |
| Frymire | 12,298 | 25386,59 | 0,164 | 160,200 | 0,8804 | 18,8180 | 157,000 | 0,831 | 20,568 |
| Barb | 0,825 | 1842,20 | 0,040 | 27,700 | 0,8813 | 29,5024 | 28,348 | 0,923 | 32,545 |
| Mandrill | 1,675 | 2841,59 | 0,040 | 45,541 | 0,8491 | 26,8805 | 76,100 | 0,943 | 32,626 |
| Crosses | 0,599 | 66,62 | 0,017 | 2,687 | 0,9286 | 21,9279 | 2,687 | 0,899 | 23,463 |
| Circles | 0,523 | 47,17 | 0,016 | 1,565 | 0,9956 | 31,5830 | 1,579 | 0,941 | 25,342 |
| Squares | 0,377 | 24,08 | 0,016 | 0,382 | 0,9999 | 62,6710 | 0,468 | 1,000 | 158,131 |
| Text | 1,946 | 384,83 | 0,050 | 21,500 | 0,9401 | 21,4640 | 28,800 | 0,998 | 31,860 |
| Horizon | 0,518 | 51,69 | 0,014 | 1,800 | 0,9524 | 27,8890 | 1,799 | 0,978 | 34,697 |

512×512), упрощая разбиение изображений неквадратного размера, в отличие от разбиения методом квадродерева. Алгоритм позволяет достичь ускорения сжатия до 1000 раз по сравнению с полным перебором, при несущественных потерях качества изображения и том же размере.

*Работа выполнена при финансовой поддержке гранта УМНИК.*

**СПИСОК ЛИТЕРАТУРЫ**

1. Уэлстид С. Фракталы и вейвлеты для сжатия изображений в действии. – М.: Триумф, 2003. – 320 с.
2. Fisher Y. Fractal Image Compression – Theory and Application. – N.Y.: Springer-Verlag, 1994. – 341 p.
3. Jacquin A. A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding: PhD Thesis. – Georgia Institute of Technology, 1989. – 138 p.
4. Hung J. Intel Wireless Display Review – No Wires? No Problem // PC Perspective. 2010. URL: http://www.pcper.com/article.php?aid=922 (дата обращения 20.02.2011).
5. Salomon D. Data Compression. The Complete Reference. 3rd edition. – N.Y.: Springer-Verlag, 2004. – 821 p.
6. Сидоров Д.В. Программный продукт imq оценки качества изображений / Оценка качества изображений. 2011. URL: http://imq.vt.tpu.ru (дата обращения 20.02.2011).
7. Test image repository / Fractal coding and analisys group. 2011. URL: http://links.uwaterloo.ca/Repository.html (дата обращения 20.02.2011).
8. Pulcini G., Verrando V. Программный продукт IFS Application Framework. 2010. URL: http://www.verrando.com/pulcini/ftp/ifsaf.zip (дата обращения 05.12.2010).

9. DeLong K. Программный продукт Fractal Image Explorer. 2009. URL: http://www.femtosoft.biz/fractals/fractalexe.zip (дата обращения 12.12.2009).
10. Sylvestre J., Fisher Y. Программный продукт FracCompress. 2010. URL: http://inls.ucsd.edu/~fisher/Fractals/Other/FracComp.zip (дата обращения 05.12.2010).
11. Barnsley M., Hurd L. Программный продукт Fractal Image Compression. URL: ftp.uu.net:/published/byte/93oct/fractal.exe (дата обращения 11.12.2009).
12. Polvere M. Программный продукт Mars. 2010. URL http://inls.ucsd.edu/~fisher/Fractals/Mars-1.0.tar.gz (дата обращения 05.12.2010).
13. Welstead S. Программный продукт IMG System. 2010. URL: http://spie.org/samples/dfractal_book.zip (дата обращения 05.12.2010).
14. Wholberg B., De Jager G. A Review of the Fractal Image Coding Literature // IEEE Trans. on Image Proc. – 1999. – V. 8. – P. 1716–1729.
15. Polvere M., Nappi M. A Feature Vector Technique for Fast Fractal Image Coding: Tech. Rep. – University of Salerno, 1998 – 86 p.
16. Hurtgen B., Stiller C. Fast Hierarchical Codebook Search for Fractal Coding of Still Image // EOS/SPIE Visual Communication and PACS for medical applications. – Berlin, 1993. – P. 397–408. DOI: 10.1117/12.160484.
17. Storer J.A., Cohn M. (eds.), Saupe D. Fractal image compression by multi-dimensional nearest neighbor search // Proc. DCC'95 Data Compression Conf. 2011. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.628&rep=rep1&type=pdf (дата обращения 21.02.2011).
18. Осокин А.Н., Шарабайко М.П. Исследование возможности распараллеливания процесса фрактального сжатия изображений // Молодежь и современные информационные технологии: Сб. трудов VIII Всеросс. научно-практ. конф. студентов, аспирантов и молодых ученых. – Томск, 2010. – Т. 1. – Ч. 2. – С. 212–213.
19. Петров А. Фрактальное сжатие графики // [Персональная страница С.А. Огрызкова]. 2010. URL: http://stanislaw.ru/rus/research/fractal.asp (дата обращения 05.12.2010).

*Поступила 15.02.2011 г.*

УДК 004

# НАХОЖДЕНИЕ ПАРАМЕТРОВ И УДАЛЕНИЕ ПОСТОЯННОЙ СОСТАВЛЯЮЩЕЙ ФИЛЬТРА ГАБОРА ДЛЯ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

А. Кермани Коланкех, В.Г. Спицын*, Ф. Хамкер

Технический университет, г. Кемниц, Германия
*Томский политехнический университет
E-mail: arash@tpu.ru

*Разработано программное обеспечение для проектирования фильтров Габора с целью обнаружения краев объектов на изображениях. Решены проблемы нахождения оптимальных параметров и удаления постоянной составляющей фильтра Габора.*

***Ключевые слова:***
*Обработка изображений, фильтр Габора, удаление DC, фильтрация, частота, гауссовский сигнал.*

***Key words:***
*Image processing, Gabor filter, Removing DC component, Filtering, Frequency, Gaussian signal.*

Фильтры Габора принадлежат к семейству полосовых фильтров [1]. Такие фильтры способны выявить диапазон частот сигнала в определенном промежутке и направлении [2], их широко используют для определения краев на изображениях.

Разработка инструментов для проектирования фильтров Габора является неотъемлемой частью многих задач обработки изображений. В данной работе использовано программное обеспечение, разработанное нами в среде Matlab для проектирования фильтров Габора, а также для учебных целей. Нам удалось найти оптимальное соотношение между частотой и шириной фильтра Габора и осуществить удаление постоянной или средней составляющей фильтра Габора – DC (*Direct Current*) компоненты.

Поскольку фильтры предназначены для обработки изображений, мы сосредоточились на 2-мерных фильтрах. Во всех экспериментах использованы фильтры 11×11 пикселей.

**Фильтр Габора**

Импульсная переходная характеристика фильтра Габора определяется в виде произведения гауссовской функции на гармоническую [3]:

$$\mathrm{Gabor}(x', y') = \mathrm{Gauss}(x', y') * \cos(2\pi f_0 x' + \varphi),$$

где

$$\mathrm{Gauss}(x) = K \frac{1}{\sqrt{2\pi\sigma_x}} e^{\left(\frac{-x^2}{2\pi\sigma_x^2}\right)} \frac{1}{\sqrt{2\pi\sigma_y}} e^{\left(\frac{-y^2}{2\pi\sigma_y^2}\right)},$$

$f_0$ и $\varphi$ – частота и фаза. Вращение фильтра на $\theta$ градусов описывается как [4]:

$$x' = x\cos\theta + y\sin\theta \quad и \quad y' = -x\sin\theta + y\cos\theta.$$

Изменяя угол вращения $\theta$, можно изменять направление, в котором необходимо обнаружить края. На рис. 1 представлены синусоидальная и гауссовская компоненты фильтра, а также сам фильтр.